Các thành phần cơ bản trong Pascal

CÁC THÀNH PHẦN CƠ BẢN CỦA

NGÔN NGỮ LẬP TRÌNH PASCAL

Pascal là một ngôn ngữ lập trình bậc cao do Niklaus Wirth, giáo sư điện toán trường Đại học kỹ thuật Zurich (Thụy Sĩ) đề xuất năm 1970. Ông lấy tên Pascal để kỷ niệm nhà toán học và nhà triết học người Pháp nổi tiếng Blaise Pascal.

1. Các tập tin cần thiết khi lập trình với Turbo Pascal

Để lập trình được với Turbo Pascal, tối thiểu cần 2 file sau:

TURBO.EXE: Dùng để soạn thảo và dịch chương trình.

TURBO.TPL: Thư viện chứa các đơn vị chuẩn để chạy với TURBO.EXE.

Ngoài ra, muốn lập trình đồ hoạ thì phải cần thêm các tập tin:

GRAPH.TPU: Thư viện đồ hoạ.

*.BGI: Các file điều khiển các loại màn hình tương ứng khi dùng đồ hoạ.

*.CHR: Các file chứa các font chữ đồ họa.

2. Các bước cơ bản khi lập một chương trình Pascal

Bước 1: Soạn thảo chương trình.

Bước 2: Dịch chương trình (nhấn phím F9), nếu có lỗi thì phải sửa lỗi.

Bước 3: Chạy chương trình (nhấn phím Ctrl-F9).

3. Cấu trúc chung của một chương trình Pascal

```
{ Phần tiêu đề }
PROGRAM  Tên_chương_trình;
{ Phần khai báo }
USES     ......;
CONST  .....;
TYPE     .......;
VAR       ........;
PROCEDURE  ...........;
FUNCTION .............;
...............
{ Phần thân chương trình }
BEGIN
     ...........
END.
```

Ví dụ 1: Chương trình Pascal đơn giản nhất

```
 Program  Vidu1;
BEGIN
   Write(‘Hello World!’);
END.
```

Ví dụ 2:

```
 Program  Vidu2;
Const  PI=3.14;
Var  R,S:Real;
Begin
   R:=10;        {Bán kính đường tròn}
   S:=R*R*PI; {Diện tích hình tròn}
   Writeln(‘Dien tich hinh tron = ‘, S:0:2);  { In ra màn hình }
   Readln;
End.
```

4. Một số phím chức năng thường dùng

F2: Lưu chương trình đang soạn thảo vào đĩa.

F3: Mở file mới hoặc file đã tồn tại trên đĩa để soạn thảo.

Alt-F3: Đóng file đang soạn thảo.

Alt-F5: Xem kết quả chạy chương trình.

F8: Chạy từng câu lệnh một trong chương trình.

Alt-X: Thoát khỏi Turbo Pascal.

Alt-<Số thứ tự của file đang mở>: Dịch chuyển qua lại giữa các file đang mở.

F10: Vào hệ thống Menu của Pascal.

5. Các thành phần cơ bản của ngôn ngữ Pascal

5.1. Từ khóa

Từ khoá là các từ mà Pascal dành riêng để phục vụ cho mục đích của nó. (Chẳng hạn như: BEGIN, END, IF, WHILE,...)

Chú ý: Với Turbo Pascal 7.0 trở lên, các từ khoá trong chương trình sẽ được hiển thị khác màu với các từ khác.

5.2. Tên (định danh)

Định danh là một dãy ký tự dùng để đặt tên cho các hằng, biến, kiểu, tên chương trình con... Khi đặt tên, ta phải chú ý một số điểm sau:

Không được đặt trùng tên với từ khoá

Ký tự đầu tiên của tên không được bắt đầu bởi các ký tự đặc biệt hoặc chữ số.

Không được đặt tên với ký tự space,các phép toán.

Ví dụ: Các tên viết như sau là sai

1XYZ Sai vì bắt đầu bằng chữ số.

#LONG Sai vì bắt đầu bằng ký tự đặc biệt.

FOR Sai vì trùng với từ khoá.

KY TU Sai vì có khoảng trắng (space).

LAP-TRINH Sai vì dấu trừ (-) là phép toán.

5.3. Dấu chấm phẩy (;)

Dấu chấm phẩy được dùng để ngăn cách giữa các câu lệnh. Không nên hiểu dấu chấm phẩy là dấu kết thúc câu lệnh.

Ví dụ:

```
FOR i:=1 TO 10 DO Write(i);
```

Trong câu lệnh trên, lệnh Write(i) được thực hiện 10 lần. Nếu hiểu dấu chấm phẩy là kết thúc câu lệnh thì lệnh Write(i) chỉ thực hiện 1 lần.

5.4. Lời giải thích

Các lời bàn luận, lời chú thích có thể đưa vào bất kỳ chỗ nào trong chương trình để cho người đọc dể hiểu mà không làm ảnh hưởng đến các phần khác trong chương trình. Lời giải thích được đặt giữa hai dấu ngoặc { và } hoặc giữa cụm dấu (* và *).

Ví dụ:

```
Var a,b,c:Real; {Khai báo biến}
```

Delta := b*b – 4*a*c; (* Tính delta để giải phương trình bậc 2 *)

BÀI TẬP THỰC HÀNH

1. Khởi động Turbo Pascal.

2. Nhập vào đoạn chương trình sau:

```
Uses Crt;

Begin

Writeln('**************************************************
******');

    Writeln('*  CHUONG TRINH PASCAL DAU TIEN CUA TOI  *');

Writeln('**************************************************
******');

    Readln;

End.
```

3. Dịch và chạy chương trình trên.

4. Lưu chương trình vào đĩa với tên BAI1.PAS.

5. Thoát khỏi Pascal.

6. Khởi động lại Turbo Pascal.

7. Mở file BAI1.PAS.

8. Chèn thêm vào dòng: CLRSCR; vào sau dòng BEGIN

9. Dịch và chạy thử chương trình.

10. Lưu chương trình vào đĩa.

11. Viết chương trình in ra màn hình:

```
        *
       ***
      **  **
     **    **
    *********
   **        **
  **          **
```

Các lệnh có cấu trúc

chương trình con

CHƯƠNG TRÌNH CON- THỦ TỤC VÀ HÀM

1. Khái niệm về chương trình con:

Chương trình con là một chương trình nằm bên trong một chương trình khác. Chương trình con có 2 loại: Thủ tục (Procedure) và hàm (Function):

- Thủ tục (PROCEDURE): Dùng để thực hiện một hay nhiều nhiệm vụ nào đó.
- Hàm (FUNCTION): Trả về một giá trị nào đó (có kiểu vô hướng, kiểu string hoặc kiểu con trỏ). Hàm có thể sử dụng trong các biểu thức.

Chương trình con được dùng rộng rãi khi xây dựng các chương trình lớn nhằm làm cho chương trình dễ theo dõi, dễ sửa chữa, có thể phân mảnh chương trình cho nhiều người làm. Một đặc điểm nổi bật của chương trình con là nó có tính đệ quy nhờ thế mà nhiều bài toán được giải quyết dễ dàng.

CẤU TRÚC CHUNG CỦA MỘT CHƯƠNG TRÌNH CÓ SỬ DỤNG CHƯƠNG TRÌNH CON:

 PROGRAM Tên_chương_trình;

USES CRT;

CONST ....;

TYPE   ....;

VAR   ....;

PROCEDURE THUTUC[(Các tham số)];

```
[Khai báo Const, Type, Var]
BEGIN
    ..............
END;
 FUNCTION  HAM[(Các tham số)]:<Kiểu dữ liệu>;
[Khai báo Const, Type, Var]
BEGIN
    ..............
    HAM:=<Giá trị>;
END;

BEGIN {Chương trình chính}
    ...................
    THUTUC[(...)];
    ...................
    A:= HAM[(...)];
    ...................
END.
```

Chú ý: Trong quá trình xây dựng CHƯƠNG TRÌNH CON, khi nào thì nên dùng thủ tục/hàm?

Dùng hàm	Dùng thủ tục
- Kết quả của bài toán trả về 1 giá trị duy nhất (kiểu vô hướng, kiểu string hoặc kiểu con trỏ).- Lời gọi CHƯƠNG TRÌNH CON cần nằm trong các biểu thức tính toán.	- Kết quả của bài toán không trả về giá trị nào hoặc trả về nhiều giá trị hoặc trả về kiểu dữ liệu có cấu trúc (Array, Record, File).- Lời gọi CHƯƠNG TRÌNH CON không nằm trong các biểu thức tính toán.

(ví dụ n!, tìm điểm đối xứng)

Ví dụ 1.1: Viết CHƯƠNG TRÌNH CON để tính n! = 1.2...n.

```
Function GiaiThua(n:integer):integer;

Var P, i:integer;

Begin

   P:=1;

   For i:=1 To n Do P:=P*i;

   GiaiThua:=P;

End;
```

Ví dụ 1.2: Viết chương trình con để tìm điểm đối xứng của điểm (x,y) qua gốc tọa độ.

```
Procedure DoiXung(x,y:Integer; Var xx,yy:Integer);
Begin
    xx:=-x;
    yy:=-y;
End;
```

CHÚ Ý: Trong 2 ví dụ trên:

n, x, y được gọi là tham trị (không có từ khóa var đứng trước) vì sau khi ra khỏi chương trình con giá trị của nó không bị thay đổi.

xx, yy được gọi là tham biến (có từ khóa var đứng trước) vì sau khi ra khỏi chương trình con giá trị của nó bị thay đổi.

1. Tham số trong chương trình con:

Các chương trình con có thể không cần tham số mà chỉ có các biến riêng (biến cục bộ). Trong trường hợp cần nhận các giá trị mà chương trình mẹ truyền cho thì chương trình con cần phải có các tham số.

Tham số thực là những giá trị lưu trữ trong các biến toàn cục của chương trình mẹ, được truyền cho các thủ tục hoặc hàm thông qua lời gọi tên của chúng.

Tham số được khai báo ngay sau tên chương trình con được gọi là tham số hình thức. Tham số hình thức gồm:

- Tham biến:

Tham biến là những giá trị mà chương trình con nhận từ chương trình mẹ, các giá trị này có thể biến đổi trong chương trình con và khi chương

trình con kết thúc các giá trị này được trả về cho tham số thực.

Cách khai báo tham biến:

Tên chương trình con (Var tên tham biến: kiểu dữ liệu);

- Tham trị:

Tham trị là những tham số truyền vào cho chương trình con xử lý nhưng khi quay về chương trình mẹ vẫn phải giữ nguyên giá trị ban đầu.

Tên chương trình con (tên tham trị: kiểu dữ liệu);

1. Truyền tham số cho chương trình con:

Khi tham số hình thức trong chương trình con là tham biến thì tham số thực trong chương trình mẹ phải là biến chứ không thể là hằng. Trong mọi trường hợp cả hai tham số thực và tham số hình thức đều phải cùng kiểu dữ liệu…(các tham biến khi ra khỏi chương trình con giá trị thay đổi).

Khi tham số hình thức là tham trị thì tham số thực phải là một giá trị.

1. Biến toàn cục và biến địa phương:

- Biến toàn cục: là các biến được khai báo trong chương trình chính. Các biến này có tác dụng ở mọi nơi trong toàn bộ chương trình.
- Biến địa phương: là các biến được khai báo trong các chương trình con. Các biến này chỉ có tác dụng trong phạm vi chương trình con đó mà thôi.

Chú ý: Trong một chương trình con, nếu biến toàn cục trùng tên với biến địa phương thì biến địa phương được ưu tiên hơn.

Ví dụ 1.3:

```
Program  KhaoSatBien;

Var a,b: Integer;   {biến toàn cục}


Procedure  ThuBien;

Var  a: Integer; {biến địa phương}

Begin

    a:=10;

    Writeln(‘A=’,a,’B=’,b);

End;


Begin

    a:=50;

    b:=200;

    ThuBien;                          {A=10  B=200}

    Writeln(‘A=’,a,’B=’,b);      {A=50  B=200}

End.
```

1. Tính đệ quy của chương trình con:

Thông thường lời gọi một chương trình con chỉ được thực hiện khi chương trình con đó đã được thiết kế hoàn chỉnh. Tuy nhiên, Pascal còn cho phép một chương trình con ngay khi trong quá trình xây dựng lại có thể gọi tới chính nó, tính chất này được gọi là “Đệ quy của chương trình con”.

1. Lời gọi chương trình con:

Một chương trình mẹ có nhiều chương trình con trực thuộc, bên trong mỗi chương trình con lại có thể có các chương trình con riêng. Khi thiết kế, mỗi chương trình con phải là một khối riêng biệt hoặc có thể có các lệnh nhảy Goto từ chương trình con này tới chương trình con khác.

- Gọi chương trình con từ trong chương trình mẹ:

Lời gọi chương trình con có thể đặt bất kỳ chỗ nào trong chương trình mẹ. Nếu chương trình con là một thủ tục thì lời gọi chương trình con có thể tạo nên một câu lệnh, ví dụ:

Readln;

Nếu chương trình con là hàm thì tên hàm không thể tạo nên một câu lệnh, vì vậy tên hàm phải nằm trong một biểu thức hay trong một thủ tục nào đó. Ví dụ, ta không thể viết:

Sqrt(9);

gọi hàm như sau là hợp lệ: a:=sqrt(9)+5;

- Gọi chương trình con từ chương trình con khác:

Các chương trình con cùng cấp có thể gọi tới nhau và truyền tham số cho nhau. Nguyên tắc gọi là: những chương trình con xây dựng sau có thể gọi tới các chương trình con đã xây dựng trước nó, đồng thời các chương trình con cấp dưới cũng có thể gọi tới các chương trình con cấp trên nếu chúng cùng một gốc. Các chương trình con xây dựng trước muốn gọi tới các chương trình con xây dựng sau thì phải có chỉ báo forward.

Xét một số ví dụ sau:

Ví dụ 1.4

Program Goi_CTC;

```
Type dayso=array[1..60] of byte; S1:=string[30];

Var

a:s1; b:dayso; i,j,n:byte;

Procedure nhapso(m:byte; var c:dayso);

Begin

For i:=1 to m do

begin

Write('c[',i.']='); readln(c[i]);

End;

End;

Function tinhtong(m:byte; var d:dayso):real;

Var tong:real;

Begin

tong:=0;

For i:=1 to m do tong:=tong+d[i];

Tinhtong:=tong;

End;

Procedure Inkq(k:byte; e: dayso);

Begin
```

Write(‘tong cac ptu =’,tinhtong(k,e):8:0); {chương trình con gọi một chương trình con cùng cấp}

Readln;

End;

BEGIN

Write(‘nhap so ptu n’); readln(n);

Nhapso(n,b);

Inkq(n,b);

End.

Nếu hàm tinhtong xây dựng sau thủ tục Inkq, thì phải có chỉ báo forward.

Thêm dòng: Function tinhtong(m:byte; var d:dayso):real; forward; trước khi xây dựng các chương trình con.

Ví dụ 1.5

Program Goi_CTC;

Type dayso=array[1..60] of byte; S1:=string[30];

Var

a:s1; b:dayso; i,j,n:byte;

Procedure nhapso(m:byte; var c:dayso);

Begin

…

End;

```
Function tinhtong(m:byte; var d:dayso):real;
Var tong:real;
Begin
…
End;
Procedure xuly(j:byte;ds:dayso);
Procedure Inkq(k:byte;e:dayso);
Var i:byte;
Begin
Writeln('tong cac phan tu mang=',tinhtong(k,e):8:0;
Writeln ('day so sap xep giam dan');
For i:=1 to k do write(e[i],' ');
Readln;
End; {ket thuc thu tuc Inkq}
Procedure sapxep(m:byte;ds:dayso);
Var p,q:byte; tg:byte;
Begin
For p:=1 to m-1 do
For q:=p+1 to m do
If(d[p]<d[q] then
```

```
Begin
Tg:=d[p];
d[p]:=d[q];
d[q]:=tg;
end;
inkq(m,d); {goi den chuong trinh con cùng cấp}
end; {ket thuc thu tuc sap xep}
Begin {than thu tuc Xuly}
Write(‘thu tuc xu ly dung de sap xep va in ket qua}
Sapxep(j,ds);
End;
BEGIN
Write(‘nhap so phan tu’); readln(n);
Nhapso(n,b);
Xuly(n,b);
END.
```

Bài tập:

Tính $C_n^k \qquad \frac{n}{k \quad n \quad k}$

(có sử dụng chương trình con)

Mảng

DỮ LIỆU KIỂU MẢNG (ARRAY)

I. KHAI BÁO MẢNG

Cú pháp:

```
TYPE    <Kiểu mảng> = ARRAY [chỉ số] OF <Kiểu dữ liệu>;
VAR     <Biến mảng>:<Kiểu mảng>;
```

hoặc khai báo trực tiếp:

```
VAR     <Biến mảng> : ARRAY [chỉ số] OF <Kiểu dữ liệu>;
```

Ví dụ:

```
TYPE    Mangnguyen = Array[1..100] of Integer;
        Matrix = Array[1..10,1..10] of Integer;
        MangKytu = Array[Byte] of Char;
VAR     A: Mangnguyen;
        M: Matrix;
        C: MangKytu;
```

hoặc:

```
VAR     A: Array[1..100] of Integer;
        C: Array[Byte] of Char;
```

II. XUẤT NHẬP TRÊN DỮ LIỆU KIỂU MẢNG

- Để truy cập đến phần tử thứ k trong mảng một chiều A, ta sử dụng cú pháp: A[k].

- Để truy cập đến phần tử (i,j) trong mảng hai chiều M, ta sử dụng cú pháp: M[i,j].

- Có thể sử dụng các thủ tục READ(LN)/WRITE(LN) đối với các phần tử của biến kiểu mảng.

BÀI TẬP MẪU

Bài tập 1: Viết chương trình tìm giá trị lớn nhất của một mảng chứa các số nguyên gồm N phần tử.

Ý tưởng:

- Cho số lớn nhất là số đầu tiên: Max:=a[1].

- Duyệt qua các phần tử a[i], với i chạy từ 2 tới N: Nếu a[i]>Max thì thay Max:=a[i];

```
Uses Crt;

Type Mang = ARRAY[1..50] Of Integer;

Var  A:Mang;

     N,i,Max:Integer;
```

```
Begin
    {Nhập mảng}
    Write('Nhap N='); Readln(N);
    For i:=1 To N Do
        Begin
            Write('A[',i,']='); Readln(A[i]);
        End;
    {Tìm phần tử lớn nhất}
    Max:=A[1];
    For i:=2 To N Do
        If Max<A[i] Then Max:=A[i];
    {In kết quả ra màn hình}
    Writeln('Phan tu lon nhat cua mang: ', Max);
    Readln;
End.
```

Bài tập 2: Viết chương trình tính tổng bình phương của các số âm trong một mảng gồm N phần tử.

Ý tưởng:

Duyệt qua tất cả các phần tử A[i] trong mảng: Nếu A[i]<0 thì cộng dồn (A[i])2 vào biến S.

```
Uses Crt;

Type Mang = ARRAY[1..50] Of Integer;

Var  A:Mang;

        N,i,S:Integer;

Begin

    {Nhập mảng}

    Write(‘Nhap N=’); Readln(N);

    For i:=1 To N Do

        Begin

           Write(‘A[‘,i,’]=’); Readln(A[i]);

        End;

    {Tính tổng}

    S:=0;

    For i:=1 To N Do

        If A[i]<0 Then S:=S+A[i]*A[i];

    {In kết quả ra màn hình}

    Writeln(‘S= ’, S);

    Readln;

End.
```

Bài tập 3: Viết chương trình nhập vào một mảng gồm N số nguyên. Sắp xếp lại mảng theo thứ tự tăng dần và in kết quả ra màn hình.

Ý tưởng:

Cho biến i chạy từ 1 đến N-1, đồng thời cho biến j chạy từ i+1 đến N: Nếu A[i]>A[j] thì đổi chổ A[i], A[j].

```
Uses Crt;

Type Mang = ARRAY[1..50] Of Integer;

Var  A:Mang;

       N,i,j,Tam:Integer;

Begin

   {Nhập mảng}

   Write('Nhap N='); Readln(N);

   For i:=1 To N Do

      Begin

         Write('A[',i,']='); Readln(A[i]);

      End;

   {Sắp xếp}

   For i:=1 To N-1 Do

      For j:=i+1 To N Do
```

```
        If A[i]>A[j] Then
           Begin
              Tam:=A[i]; A[i]:=A[j]; A[j]:=Tam;
           End;
   {In kết quả ra màn hình}
   Writeln('Ket qua sau khi sap xep:');
   For i:=1 To N Do Write(A[i]:5);
   Readln;
End.
```

Bài tập 4: Viết chương trình nhập vào một mảng A gồm N số nguyên và nhập thêm vào một số nguyên X. Hãy kiểm tra xem phần tử X có trong mảng A hay không?

Ý tưởng:

Dùng thuật toán tìm kiếm tuần tự. So sánh x với từng phần tử của mảng A. Thuật toán dừng lại khi x=A[i] hoặc i>N.

Nếu x=A[i] thì vị trí cần tìm là i, ngược lại thì kết quả tìm là 0 (không tìm thấy).

```
Uses Crt;
Type Mang = ARRAY[1..50] Of Integer;
Var  A:Mang;
```

```
        N,i,x:Integer;

Function TimKiem(x, N: Integer; A:Mang):Integer;
Var i:Integer;
    Begin
        I:=1;
        While (I <= N) and (X<>A[I]) do I:=I+1;
        If I <= N Then Timkiem:=I  Else Timkiem:=0;
    End;

Begin
    {Nhập mảng}
    Write(‘Nhap N=’); Readln(N);
    For i:=1 To N Do
        Begin
           Write(‘A[‘,i,’]=’); Readln(A[i]);
        End;
    Write(‘Nhap X=’); Readln(x);
    {Kết quả tìm kiếm}
    If TimKiem(X,N,A)<>0 Then
```

```
      Writeln(‘Vi tri cua X trong mang la:’, TimKiem(X,N,A))

   Else Writeln(‘X khong co trong mang.’);

   Readln;

End.
```

Bài tập 5: Giả sử mảng A đã được sắp xếp theo thứ tự tăng dần. Viết hàm để kiểm tra xem phần tử X có trong mảng A hay không?

Ý tưởng:

So sánh x với phần tử ở giữa mảng A[giua]. Nếu x=A[giua] thì dừng (vị trí cần tìm là chỉ số của phần tử giữa của mảng). Ngược lại, nếu x>A[giua] thì tìm ở đoạn sau của mảng [giua+1,cuoi], ngược lại thì tìm ở đoạn đầu của mảng [dau,giua-1].

Sau đây là hàm cài đặt cho thuật toán này:

```
Function TimKiemNhiPhan(X, N: Integer; A: Mang):Integer;

Var  dau,cuoi,giua:Integer;

      Found:Boolean;

Begin

   dau:=1; {điểm mút trái của khoảng tìm kiếm}

   cuoi:=N; {điểm mút phải của khoảng tìm kiếm}

   Found:=False; {chưa tìm thấy}

   While (dau <=cuoi) and (Not Found) Do

      Begin
```

```
        giua:=(dau + cuoi) Div 2;
        If  X = A[giua] Then Found:=True {đã tìm thấy}
        Else
            If X > A[giua] Then dau:=giua+1
            Else cuoi:=giua-1;
      End;
   If Found Then TimKiemNhiPhan:= giua Else TimKiemNhiPhan:=0;
End;
```

Bài tập 6: Cho một mảng 2 chiều A cấp mxn gồm các số nguyên và một số nguyên x. Viết chương trình thực hiện các công việc sau:

a/ Đếm số lần xuất hiện của x trong A và vị trí của chúng.

b/ Tính tổng các phần tử lớn nhất của mỗi dòng.

```
Uses Crt;
Type Mang = ARRAY[1..10,1..10] Of Integer;
Var  A:Mang;
      m,n,i,j,x,dem,S,max:Integer;
Begin
   {Nhập ma trận}
```

```
Write('Nhap số dòng m='); Readln(m);

Write('Nhap số cột n='); Readln(n);

For i:=1 To m Do

    For j:=1 To n Do

       Begin

           Write('A[',i,j,']='); Readln(A[i,j]);

       End;

{Nhập x}

Write('Nhap x='); Readln(x);

{Đếm số lãn xuất hiện của x và vị trí của x}

dem:=0;

Writeln('Vi tri cua x trong mang A: ');

For i:=1 To m Do

    For j:=1 To n Do

       If x=A[i,j] Then

           Begin

               Write(i,j,' ; ');

               dem:=dem+1;

           End;

Writeln('So lan xuat hien cua x trong mang A la: ',dem);
```

```
    {Tính tổng các phần tử lớn nhất của mỗi dòng}
    S:=0;
    For i:=1 To m Do {duyệt qua từng dòng}
      Begin
        {Tìm phần tử lớn nhất của dòng thứ i}
        Max:=A[i,1];
        For j:=2 To n Do   {duyệt từng phần tử của dòng thứ i}
          If max<A[i,j] Then max:=A[i,j];
        {Cộng max vào biến S}
        S:=S+max;
      End;
    Writeln('Tong cac phan tu lon nhat cua moi dong la: ',S);
    Readln;
End.
```

Xâu ký tự

XÂU KÝ TỰ (STRING)

I. KHAI BÁO KIỂU STRING

TYPE   TênKiểu = STRING[Max];

VAR   Tên biến : TênKiểu;

hoặc khai báo biến trực tiếp:

VAR   Tên biến : STRING[Max];

Trong đó Max là số ký tự tối đa có thể chứa trong chuỗi (Max Î [0,255]). Nếu không có khai báo [Max] thì số ký tự mặ mặc định trong chuỗi là 255.

Ví dụ:

```
Type      Hoten  = String[30];
          St80 = String[80];
Var       Name : Hoten;
          Line : St80;
          St : String; {St có tối đa là 255 ký tự}
```

II. TRUY XUẤT DỮ LIỆU KIỂU STRING

- Có thể sử dụng các thủ tục xuất nhập Write, Writeln, Readln để truy xuất các biến kiểu String.

- Để truy xuất đến ký tự thứ k của xâu ký tự, ta sử dụng cú pháp sau: Tênbiến[k].

III. CÁC PHÉP TOÁN TRÊN XÂU KÝ TỰ

3.1. Phép nối xâu: +

3.2. Các phép toán quan hệ: =, <>, <, <=, >, >=.

Chú ý: Các phép toán quan hệ được so sánh theo thứ tự từ điển.

IV. CÁC THỦ TỤC VÀ HÀM VẾ XÂU KÝ TỰ

4.1. Hàm lấy chiều dài của xây ký tự

LENGTH(St : String):Integer;

4.2. Hàm COPY(St : String; Pos, Num: Byte): String;

Lấy ra một xâu con từ trong xâu St có độ dài Num ký tự bắt đầu từ vị trí Pos .

4.3. Hàm POS(SubSt, St :String):Byte;

Kiểm tra xâu con SubSt có nằm trong xâu St hay không? Nếu xâu SubSt nằm trong xâu St thì hàm trả về vị trí đầu tiên của xâu con SubSt trong xâu St, ngược lại hàm trả về giá trị 0.

4.4. Thủ tục DELETE(Var St:String; Pos, Num: Byte);

Xoá trong xâu St Num ký tự bắt đầu từ vị trí Pos.

4.5. Thủ tục INSERT(SubSt: String; Var St: String; Pos: Byte);

Chèn xâu SubSt vào xâu St bắt đầu tại vị trí Pos.

4.6. Thủ tục STR(Num; Var St:String);

Đổi số nguyên hay thực Num thành dạng xâu ký tự, kết quả lưu vào biến St.

4.7. Thủ tục VAL(St:String; Var Num; Var Code:Integer);

Đổi xâu số St thành số và gán kết quả lưu vào biến Num. Nếu việc chuyển đổi thành công thì biến Code có giá trị là 0, ngược lại biến Code có giá trị khác 0 (vị trí của lỗi).

BÀI TẬP MẪU

Bài tập 1: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Đổi xâu ký tự đó sang chữ in hoa rồi in kết quả ra màn hình.

Ví dụ :Xâu abcdAbcD sẽ cho ra xâu ABCDABCD.

```
Uses Crt;

Var  St:String;

     i:Byte;

Begin

   Write('Nhap xau St: '); Readln(St);

   For i:=1 to length(St) do St[i]:=Upcase(St[i]);
```

```
    Write('Xau ket qua: ', St);

    Readln;

End.
```

Bài tập 2: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Đổi xâu ký tự đó sang chữ thường rồi in kết quả ra màn hình.

Ví dụ :Xâu abCdAbcD sẽ cho ra xâu abcdabcd.

```
Uses Crt;

Var  St:String;

        i:Byte;

Begin

    Write('Nhap xau St: '); Readln(St);

    For i:=1 to length(St) do

        If St[i] IN ['A'..'Z'] Then St[i]:=CHR(ORD(St[i])+32);

    Write('Xau ket qua: ', St);

    Readln;

End.
```

Bài tập 3: Viết chương trình đếm số ký tự chữ số trong một xâu ký tự được nhập vào từ bàn phím.

```
Uses Crt;

Var  St:String;

        i,d:Byte;

Begin

    Write('Nhap xau St: '); Readln(St);

    For i:=1 to length(St) do

        If St[i] IN ['0'..'9'] Then d:=d+1;

    Write('So ky tu chu so trong xau: ', d);

    Readln;

End.
```

Bài tập 4: Viết chương trình nhập một xâu từ bàn phím. In ra xâu đó sau khi xóa hết các ký tự trắng thừa trong xâu. (Ký tự trắng thừa là các ký tự trắng đầu xâu, cuối xâu và nếu ở giữa xâu có 2 ký tự trắng liên tiếp nhau thì có 1 ký tự trắng thừa).

```
Uses Crt;

Var  St:String;

Procedure  XoaTrangThua(Var St:String);

Begin
```

```
    {Xóa các ký tự trắng ở đầu xâu}
    While St[1]=#32 Do Delete(St,1,1);
    {Xóa các ký tự trắng ở cuối xâu}
    While St[Length(St)]=#32 Do Delete(St,Length(St),1);
    {Xóa các ký tự trắng ở giữa xâu}
    While POS(#32#32,St)<>0 Do Delete(St,POS(#32#32,St),1);
End;

Begin
    Write(‘Nhap xau St: ‘); Readln(St);
    XoaTrangThua(St);
    Write(‘Xau sau khi xoa cac ky tu trang thua: ‘, St);
    Readln;
End.
```

Bài tập 5: Viết chương trình liệt kê các từ của một xâu ký tự được nhập vào từ bàn phím, mỗi từ phải được viết trên một dòng.

```
Uses Crt;
Var  St:String;
```

```
Procedure  XoaTrangThua(Var St:String);
Begin
    {Xóa các ký tự trắng ở đầu xâu}
    While St[1]=#32 Do Delete(St,1,1);
    {Xóa các ký tự trắng ở cuối xâu}
    While St[Length(St)]=#32 Do Delete(St,Length(St),1);
    {Xóa các ký tự trắng ở giữa xâu}
    While POS(#32#32,St)<>0 Do Delete(St,POS(#32#32,St),1);
End;

Begin
    Write(‘Nhap xau St: ‘); Readln(St);
    XoaTrangThua(St);
    St:=St+#32;
    Writeln(‘Liet ke cac tu trong xau: ‘);
    While POS(#32,St)<>0 Do
        Begin
           Writeln(Copy(St,1,POS(#32,St)));
           Delete(St,1,POS(#32,St));
        End;
```

```
    Readln;
End.
```

Bài tập 6: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Tìm xâu đảo ngược của xâu đó rồi in kết quả ra màn hình theo 2 cách: Đệ qui và không đệ qui.

Ý tưởng:

- Nếu xâu St có 1 ký tự thì xâu đảo = St.
- Ngược lại: Xâu đảo = Ký tự cuối + Đệ qui(Phần còn lại của xâu St).

```
Uses Crt;
Var  St:String;

{Giải thuật không đệ qui}
Function XauDao(St:String):String;
Var  S:String;
       i:Byte;
Begin
    S:='';
    For i:=Length(St) DowTo 1 Do S:=S+St[i];
    XauDao:=S;
```

```
End;

{Giải thuật đệ qui}
Function DeQui(St:String):String;
Begin
   If Length(St)<=1 Then DeQui:=St
   Else DeQui:=St[Length(St)] + DeQui(Copy(St,1,Length(St)-1));
End;

Begin
   Write('Nhap xau St: '); Readln(St);
   Write('Xau dao nguoc: ', XauDao(St));
   Readln;
End.
```

Bài tập 7: Viết chương trình nhập vào một xâu ký tự từ bàn phím. Thông báo lên màn hình các chữ cái có trong xâu và số lượng của chúng (Không phân biệt chữ hoa hay chữ thường).

Ý tưởng:

- Dùng một mảng dem với chỉ số là các chữ cái để lưu trữ số lượng của các chữ cái trong xâu.

- Duyệt qua tất cả các ký tự của xâu St: Nếu ký tự đó là chữ cái thì tăng ô biến mảng dem[St[i]] lên 1 đơn vị.

```
Uses Crt;
Var  St:String;
        dem: Array[‘A’..’Z’] Of Byte;
        i:Byte;
        ch:Char;
Begin
    Write(‘Nhap xau St: ‘); Readln(St);
    {Khởi tạo mảng}
    For ch:=’A’ To ‘Z’ Do dem[ch]:=0;
    {Duyệt xâu}
    For i:=1 To Length(St) Do
        If Upcase(St[i]) IN [‘A’..’Z’] Then Inc(dem[Upcase(St[i])]);
    {Liệt kê các ký tự ra màn hình}
    For ch:=’A’ To ‘Z’ Do
        If dem[ch]>0 Then Writeln(ch,’ : ’,dem[ch]);
    Readln;
End.
```

Bài tập 8: Viết chương trình xóa các ký tự chữ số trong một xâu ký tự được nhập vào từ bàn phím.

```
Uses Crt;

Var  St:String;


{Hàm POSNUM kiểm tra xem trong xâu St có ký tự chữ số hay không? Nếu có, hàm trả về vị trí đầu tiên của ký tự chữ số, ngược lại hàm trả về giá trị 0}

Function POSNUM(St:String):Byte;

Var  OK:Boolean;

        i:Byte;

Begin

    OK:=False;

    i:=1;

    While (i<=Length(St)) AND (Not OK) Do

        If St[i] IN [‘0’..’9’] Then OK:=True

        Else i:=i+1;

    If OK Then POSNUM:=i Else POSNUM:=0;

End;
```

```
Begin
    Write('Nhap xau St: '); Readln(St);
    While POSNUM(St)<>0 Do Delete(St,POSNUM(St),1);
    Write('Xau sau khi xoa: ',St);
    Readln;
End.
```

Bài tập 9: Viết chương trình để mã hoá và giải mã một xâu ký tự bằng cách đảo ngược các bit của từng ký tự trong xâu.

```
Uses crt;
Var st:string;

{Hàm đảo bit ký tự c}
Function DaoBit(c:char):char;
Var n,i,s,bitcuoi,Mask:byte;
Begin
 {Đổi ký tự sang số}
  n:=ORD(c);
```

```
 {s: kết quả đảo bit, Mask: mặt nạ dùng để bật bit thứ i}
  s:=0;
  Mask:=128;

  For i:=1 To 8 Do {duyệt qua 8 bit của n}
   Begin
     {Lấy bit cuối cùng của n: bit cực phải}
     bitcuoi:=n AND 1;
     n:=n shr 1; {loại bỏ bit cuối cùng: n:=n DIV 2}
     {Bật bit thứ i lên: từ trái sang phải}
     if bitcuoi=1 then s:=s OR Mask;
     Mask:=Mask shr 1; { Mask:= Mask DIV 2}
   End;

  DaoBit:=CHR(s);
End;

Function MaHoa(st:string):string;
Var i:Byte;
Begin
```

```
 {Đảo bit từng ký tự trong xâu st}
  For i:=1 To Length(st) Do st[i]:=DaoBit(st[i]);
  Mahoa:=st;
End;

Begin
 Write('Nhap xau: '); Readln(st);
 st:=MaHoa(st);
 Writeln('Xau sau khi ma hoa: ',st);
 Readln;
 st:=MaHoa(st);
 Writeln('Xau sau khi giai ma: ',st);
 Readln;
End.
```

Bài tập 10: Viết chương trình thực hiện phép cộng 2 số tự nhiên lớn (không quá 255 chữ số).

```
Uses crt;
Var so1,so2,kqua:string;
```

```
Procedure LamDayXau(Var st1,st2:string);
{Them so 0 vao truoc xau ngan}
var i:Byte;
Begin
  If Length(st1)>Length(st2) Then
    For i:=1 To Length(st1)-Length(st2) Do st2:='0'+st2
  Else
    For i:=1 To Length(st2)-Length(st1) Do st1:='0'+st1;
End;

Function Cong(st1,st2:string):string;
Var i,a,b,c,sodu:Byte;
    code:integer;
    st,ch:string;
Begin
  st:=''; sodu:=0;
  LamDayXau(st1,st2);
  {Lấy từng số của 2 xâu: từ phải sang trái}
  For i:=Length(st1) DownTo 1 Do
```

```
Begin
  {Đổi ký tự sang số nguyên}
  Val(st1[i],a,code);
  Val(st2[i],b,code);
  {Tính tổng của 2 số a,b vừa lấy ra cho vào biến c}
  c:=(a+b+sodu) MOD 10;
  {Lấy phần dư của tổng a+b}
  sodu:=(a+b+sodu) DIV 10;
  {Đổi số nguyên c sang xâu ký tự ch}
  str(c,ch);
  {Cộng xâu ch vào bên trái xâu kết quả st}
  st:=ch+st;
End;

 {Xử lý trường hợp số dư cuối cùng >0}
If sodu>0 Then
 Begin
  str(sodu,ch);
  st:=ch+st;
 End;
```

```
   Cong:=st;
End;

Begin
  Write('Nhap so thu nhat: '); Readln(so1);
  Write('Nhap so thu hai: '); Readln(so2);
  kqua:=Cong(so1,so2);
  Writeln('Tong= ',kqua);
  Readln;
End
```

Đệ quy

Dữ liệu kiểu tệp

KIỂU DỮ LIỆU TỆP

1 Khái niệm về tệp:

Tệp là một dãy các phần tử cùng kiểu được sắp xếp một cách tuần tự. Tệp dữ liệu được lưu trữ ở bộ nhớ ngoài dưới một tên nào đó.

Tệp tập hợp trong nó một số phần tử dữ liệu có cùng cấu trúc giống như mảng nhưng khác mảng là số phần tử của tệp chưa được xác định.

Trong Pascal có 3 loại tệp được sử dụng là:

1. Tệp có kiểu:

Tệp có kiểu là tệp mà các phần tử của nó có cùng độ dài và cùng kiểu dữ liệu.

1. Tệp văn bản:

Dùng để lưu trữ dữ liệu dưới dạng các ký tự của bảng mã ASCII, các ký tự này được lưu thành từng dòng, độ dài các dòng có thể khác nhau. Ví dụ 2008 (kiểu word) khi ghi vào tệp văn bản cần 4 Byte (không phải 2 Byte).

1. Tệp không kiểu:

Tệp không kiểu là một loại tệp không cần quan tâm đến kiểu dữ liệu ghi trên tệp. Dữ liệu ghi vào tệp không cần chuyển đổi.

Tác dụng lớn nhất của kiểu dữ liệu tệp là ta có thể lưu trữ các dữ liệu nhập vào từ bàn phím và các kết quả xử lý trong bộ nhớ RAM ra tệp để dùng nhiều lần.

1. Khai báo:

- Định nghĩa kiểu tệp với từ khóa FILE OF trong phần mô tả kiểu sau từ TYPE, tiếp theo là khai báo biến tệp trong phần khai báo biến.

Ví dụ 2.6:

Type

MSN=Array[1..100] of integer; {định nghĩa mảng 100 số nguyên}

TSN= File of MSN; {định nghĩa tệp TSN có các phần tử là mảng số nguyên}

TCV=File of String[80]; {định nghĩa tệp TCV có các phần tử là các chuỗi có độ dài 80 ký tự.

Bangdiem= Record

……

End;

TBD= File of Bangdiem;

Var:

Tep1: TSN;

Tep2: TCV;

Tep3: TBD;

- Định nghĩa trực tiếp biến kiểu tệp trong phần khai báo biến

Var

Tep4:File of Array[1..5] of String[80];

Tep5: File of Bangdiem;

1. Truy nhập vào tệp:

Turbo Pascal có thể xử lý 2 loại tệp là : Tệp truy nhập tuần tự và tệp truy nhập trực tiếp.

- Tệp truy nhập tuần tự: để truy nhập vào một phần tử nào đó, ta bắt buộc phải đi qua các phần tử trước đó. Nếu muốn thêm các phần tử vào tệp thì có thể thêm vào cuối tệp.
- Tệp truy nhập trực tiếp: là tệp có thể truy nhập vào phần tử bất kỳ trong tệp. Muốn truy nhập trực tiếp phải dùng thủ tục Seek (số hiệu phần tử).
- Mở tệp:

Để mở một tệp chuẩn bị lưu trữ dữ liệu, ta sử dụng 2 thủ tục chuẩn sau đây:

ASSIGN(biến tệp, tên tệp);

REWRITE(biến tệp);

Trong đó:

Biến tệp: là tên biến tệp đã khai báo sau từ khóa VAR

Tên tệp: Là tên do ta chọn để ghi dữ liệu vào đĩa.

Ví dụ : ASSIGN(f, ‘a:\baitap.txt’);

REWRITE(f); {khởi tạo tệp rỗng}

Sau 2 thủ tục trên, để tiến hành ghi dữ liệu vào tệp ta lại dùng thủ tục WRITE(…):

Cách viết:

WRITE(biến tệp, các giá trị cần ghi vào tệp);

Cuối cùng, ta phải đóng tệp bằng thủ tục:

CLOSE(biến tệp);

2 Tệp văn bản:

a. Khai báo tệp văn bản:

Tệp văn bản được khai báo trực tiếp trong phần khai báo biến:

Var

Bientep:Text;

b. Truy nhập vào tệp:

Truy nhập vào tệp được hiểu là nhập dữ liệu vào tệp, ghi lại dữ liệu trên thiết bị nhớ ngoài, đọc dữ liệu đó ra màn hình hoặc máy in và xử lý nó.

- Mở tệp mới để ghi:

Assign(bientep, tentep);

Rewrite(bientep);

- Mở tệp đã có để ghi thêm:

Assign(bientep, tentep);

Append(bientep);

- Mở tệp để đọc dữ liệu:

Assign(bientep, tentep);

Reset(bientep);

c. Ghi dữ liệu vào tệp:

Sau khi đã mở tệp chúng ta có thể dùng thủ tục Write hoặc Writeln để ghi dữ liệu vào tệp.

Ví dụ 2.7:

```
Var
T1:Text;
Begin
Assign(T1,'Dulieu.dat');
Rewrite(T1);
Writeln(T1,'Tep van ban');
Write(T1,123);
Write(T1,' ',123.45);
Writeln(T1);
Close(T1);
End.
```

Dữ liệu ghi vào tệp như sau:

Tep van ban

123 1.234500000E+02

Dòng trống

d. Đọc dữ liệu từ tệp văn bản:

Sau khi tiến hành mở tệp, con trỏ tệp sẽ được đặt tại dòng đầu.Ta dùng thủ tục Read hoặc Readln để đọc dữ liệu từ dòng hiện thời và gán vào biến tương ứng, viết biến đó ra màn hình hoặc máy in.

Để có thể viết toàn bộ dữ liệu từ một tệp văn bản ra các thiết bị ngoài thì, thì các lệnh đọc viết phải được lặp đi lặp lại từ dòng 1 đến dòng cuối cùng, nghĩa là phải sử dụng một trong 2 vòng lặp:

While not eof(Bientep) do

Begin

Readln(Bientep, Dong); {biến Dong phải được khai báo trước, kiểu String}

Write(Dong);

End;

Hoặc:

For i:=1 to filesize(Bientep) do

Begin

Readln(Bientep,Dong);

Write(Dong);

End;

Lưu ý: Muốn lấy lại kiểu của dữ liệu nhập vào tệp văn bản thì mỗi biến phải nhập trên một dòng.

Ví dụ 2.8:

Xây dựng một chương trình đơn giản để quản lý công chức. Dữ liệu nhập bao gồm: Họ tên, Hệ số lương và số con. Dữ liệu xuất ra màn hình bao gồm Họ tên, Hệ số lương, Số con và Lương tháng (tính theo quy định của nhà nước = heso*540000).

Chương trình đặt ra hai khả năng lựa chọn:

1. Nếu tệp dữ liệu đã tồn tại thì nhập thêm người
2. Nếu tệp chưa có thì mở tệp mới

Trong cả 2 trường hợp đều cho biết số người cần nhập. Dữ liệu in ra dưới dạng bảng.

Program Quan_ly_can_bo;

Uses Crt;

Var f:Text; hoten:String[20]; c1, heso:real; c2,i,n,socon:byte;

Ten:string[12];

Begin

Clrscr;

Write(‘cho biet ten tep’); readln(ten);

Assign(f,ten);

Reset(f);

If IOResult=0 then

Append(f);

Else

Rewrite(f);

Write(‘nhap bao nhieu nguoi’); readln(n);

For i:=1 to n do

Begin

Write(‘Hoten’); Readln(hoten);

```
Write(‘He so’); Readln(heso);

Write(‘So con’); Readln(socon);

Writeln(f,hoten);

Writeln(f,heso:4:2);

Writeln(f,socon);

End;

Close(f);

Assign(f,ten);

Reset(f);

Writeln(‘----------------------------------------------------‘);

Writeln (‘| Ho va ten | Hs | socon | Luong |’);

Writeln(‘----------------------------------------------------‘)

While not eof(f) do

Begin

Readln(f,hoten);

Readln(f,heso);

Readln(f,socon);

Writeln(‘|’, hoten:19,’|’,heso:4:2,’|’,socon:4,’|’,heso*540000:10:2,’|’);

End;

Readln;
```

End.

3 Tệp có kiểu:

a. Đọc và ghi :

- Ghi lên tệp:

Write(bientep,bien1,bien2,…);

bien1,bien2,…là các biến cùng kiểu với biến tệp.

- Đọc tệp:

Read(bientep,bien1,bien2,…);

Chú ý:

Khác với tệp văn bản, việc ghi và đọc tệp có kiểu không sử dụng các lệnh Writeln hoặc readln nghĩa là tệp có kiểu không ghi dữ liệu thành các dòng. Các phần tử của tệp có kiểu được ghi liên tục trong các ô nhớ và chỉ có ký hiệu kết thúc tệp EOF.

Khi chúng ta đọc hoặc ghi xong một phần tử thì con trỏ tệp sẽ tự động chuyển đến vị trí kế tiếp.

1. Truy nhập vào tệp:

Seek(bientep,i); i=0,1,2,…

Thủ tục seek sẽ định vị con trỏ tại vị trí thứ i của tệp.

1. các hàm xử lý tệp:

- Filesize(bientep) cho biết số phần tử có trong tệp
- FilePos(bientep) cho biết vị trí hiện thời của con trỏ tệp
- Eof(Bientep) cho giá trị là True nếu con trỏ tệp ở vị trí cuối tệp, ngược lại cho giá trị False

Ví dụ 2.9:

Tạo một tệp lấy tên là TEPCK.DAT để vừa ghi vừa sửa dữ liệu:

```
Program Tep_co_kieu:

Uses crt;

Var bt:file of byte; i:byte; n:real;

Begin

Clrscr;

Assign(bt,’ TEPCK.DAT’);

Rewrite(bt);

For i:=0 to 5 do write(bt,i); {ghi vào tệp 5 số nguyên}

Reset(bt);

Writeln(‘Du lieu luu tru trong tep TEPCK.DAT’);

While not eof(BT) do

Begin

Read(bt,i); write(i:5);

End;

Writeln;

Seek(bt,3); {định vị con trỏ tại phần tử thứ 4}

Textcolor(magenta);

Read(bt,i);
```

Writeln (‘So trong tep o vi trí thu 4:’,i);

i:=33;

seek(bt,3);

write(bt,i);

seek(bt,3); read(bt,i);

writeln(‘So moi trong tep o vi tri 4:,i);

writeln(‘vi tri hien thoi cua con tro:’, filepos(bt));

readln;

close(bt);

end.

4 Tệp không kiểu:

a. Khai báo biến tệp:

Var Bientep:File;

b. Mở tệp để ghi-đọc:

- Mở tệp mới để ghi:

Assign(bientep, tentep);

Rewrite(bientep, n);

- Mở tệp để đọc dữ liệu:

Assign(bientep, tentep);

Reset(bientep, n);

Với n là độ lớn tính theo Byte.

c. Đọc và ghi tệp không định kiểu:

* Đọc tệp không định kiểu:

BlockRead(bientep,biennho,i,j);

- biennho: là biến đã được khai báo cùng kiểu với các phần tử của tệp, biến nhớ đóng vai trò vùng nhớ đệm để lưu trữ dữ liệu đọc từ phần tử của tệp ra.
- i: là số phần tử quy định cho mỗi lần đọc.
- j: là biến kiểu Word, dùng để ghi lại số phần tử thực sự đã được đọc.

* Ghi tệp không định kiểu:

BlockWrite(bientep,biennho,i);

1. Truy nhập tệp không định kiểu:

Tệp không kiểu cũng được truy nhập như tệp có kiểu nghĩa là cũng dùng thủ tục Seek(bientep,n) để truy nhập vào phần tử thứ n+1 của tệp.

Lưu ý là với tệp không kiểu, mỗi lần con trỏ dịch chuyển nó sẽ dịch chuyển một số byte đúng bằng số byte đã quy định trong lệnh Rewrite() hoặc Reset()

Ví dụ 2.10

Nhập vào tệp các phần tử là record và sau đó viết chúng ra màn hình. Trong phần khai báo record chọn Hoten là string[15] và Diem thuộc kiểu Real.

Program tep_khong_kieu;

Uses Crt;

Type hs=record

Hoten:string[15];

Diem:real;

End;

Var

bt:file; k,nguoi:hs; i,j:byte;

```
Begin
Clrscr;
assign(bt,'tep0kieu.dat');
rewrite(bt,22);
write('Nhap bao nhieu nguoi? ');
readln(n);
   For i:= 1 to n  do
 with nguoi do
 Begin
 write('Ho va ten : '); readln(hoten);
 Write('Diem tong : '); readln(diem);
 blockwrite(bt,nguoi,1);
end;
for i:= 0 to n-1 do
begin
seek(bt,i);
Blockread(bt,k,1);
textcolor(red);
with k do
writeln(hoten,'  ',diem:5:2);
end;
```

Close(bt);

```
Readln;

End.
```